Algoritmos

... e computadores

- Algoritmo:

  programa, software ...

- Computador, HD, disquete, ... :

  hardware, executores, atores

- Entrada:

  teclado, mouse, sensores, ...

- Saída:

  monitor, impressora, ...

Algoritmos

... e computadores

Características dos algoritmos como software:

1. Texto finito:

   todo programa tem um texto (talvez muitas linhas) finito .....

2. Instruções elementares:

   elementares para o computador onde o software vai executar.

   Dificuldades:

   - Cada computador tem um **particular** conjunto de instruções básicas
   - Instruções do computador são **muito primitivas**

   Solução:escrever algoritmos em uma **linguagem de programação** (**C**, **C++**, **JAVA**, **FORTRAN**, . . .)

   ⇒ **Programa (software): texto** escrito numa particular **LP**

Algoritmos

... e computadores

Características dos algoritmos como software (cont):

3. Receita metódica:

   texto escrito numa LP é **preciso** e **sem ambigüidades**

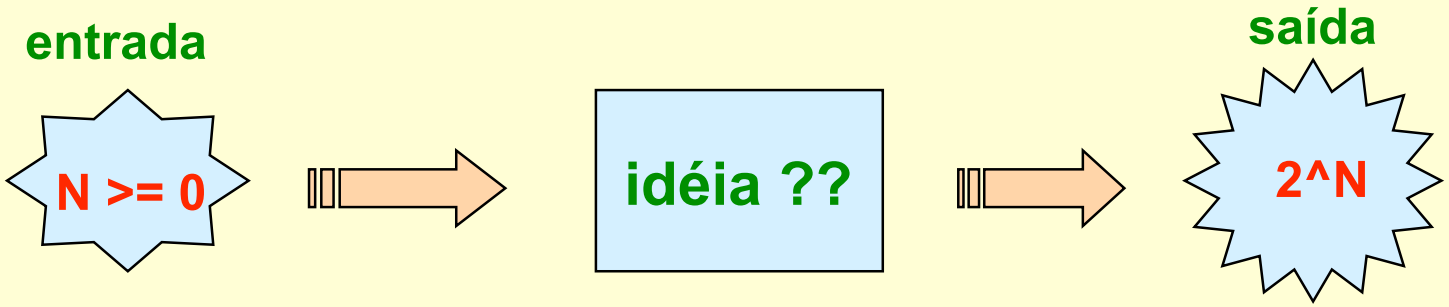
4. Terminação:
   1. **Grande desafio**: texto escrito numa LP **não deixa isso claro**.
   2. **Problema** no desenvolvimento de software: execução sem terminação (i.e. sem “loops”).

Algoritmos

... e computadores

Exemplo:

Problema: dado um número N >= 0, calcular 2^N



Algoritmos

... e computadores

Exemplo (cont):

Idéia:

$2^N = 1 \underbrace{x2x2x \ldots x2}_{N \text{ vezes}}$ (quando N = 0, 2^0 = 1)

Algoritmo:

1. Copie o valor de N para Z
2. Copie o valor 1 para P
3. Enquanto (Z > 0) faça
   - 3.1 Novo valor de P é 2x(valor atual de P)
   - 3.2 Novo valor de Z é (valor atual de Z) - 1
4. Resposta está em P; páre

Algoritmos

... e computadores

Exemplo (cont):

Escrevendo texto numa LP, p. ex. em C

```
. . . .
Z = N;
P = 1;
while (Z>0) do
 {
    P = 2xP;
    Z = Z – 1;
 }
 . . . .
```

1. Copie o valor de N para Z
2. Copie o valor 1 para P
3. Enquanto (Z > 0) faça
   3.1 Novo valor de P é
   2x(valor atual de P)
   3.2 Novo valor de Z é
   (valor atual de Z) - 1
4. Resposta está em P; páre

Algoritmos

... e compilação

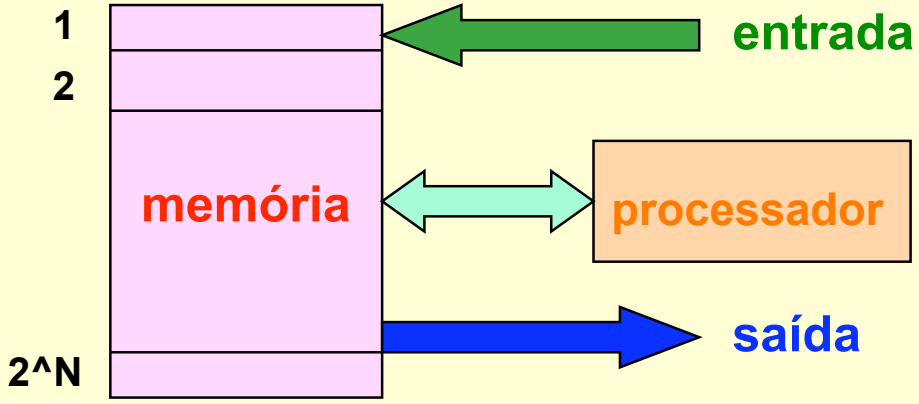
Um computador é, essencialmente, uma máquina programável muito "primitiva"

- **instruções elementares** (de máquina) primitivas
- lidam apenas com **pequenas cadeias** de "bits"
- realizam **operações muito simples** sobre essas cadeias de bits
  - **trocar um bit** (de 0 para 1, ou de 1 para 0), dependendo do valor atual de outro bit
  - **armazenar na memória** uma cadeia de bits
  - **recuperar da memória** uma cadeia de bits

Algoritmos

... e compilação

Arquitetura básica simplificada:



Memória

N	2^N	Bytes
10	1024	1 K
20	1048576	1 M
27	134217728	128 M
30	1073741824	1 G
32	4294967296	4 G

1 byte = 8 bits

Algoritmos

... e compilação

Arquitetura básica simplificada (cont):

- Formado por memória (grande), processador e circuitos de entrada e de saída
- Processador executa instruções elementares **específicas dessa máquina**
  - inclusive armazenamento e recuperação de dados da memória
- Processador e circuitos de e/s **recebem dados** das entradas e **exibem dados** nas saídas

Algoritmos

... e compilação

Compilação:

- Processo de traduzir programas escritos em uma particular LP para código em instruções básicas de uma máquina específica
- Tradução de LP para linguagem de máquina:
  - **Dificuldades**: processo laborioso, entediante e sujeito a erros.
  - **Solução**: Escrever um programa para fazer a tradução

→ Esse programa é um **compilador**

Algoritmos

... e compilação

Para cada LP e cada computador (processador), um compilador específico

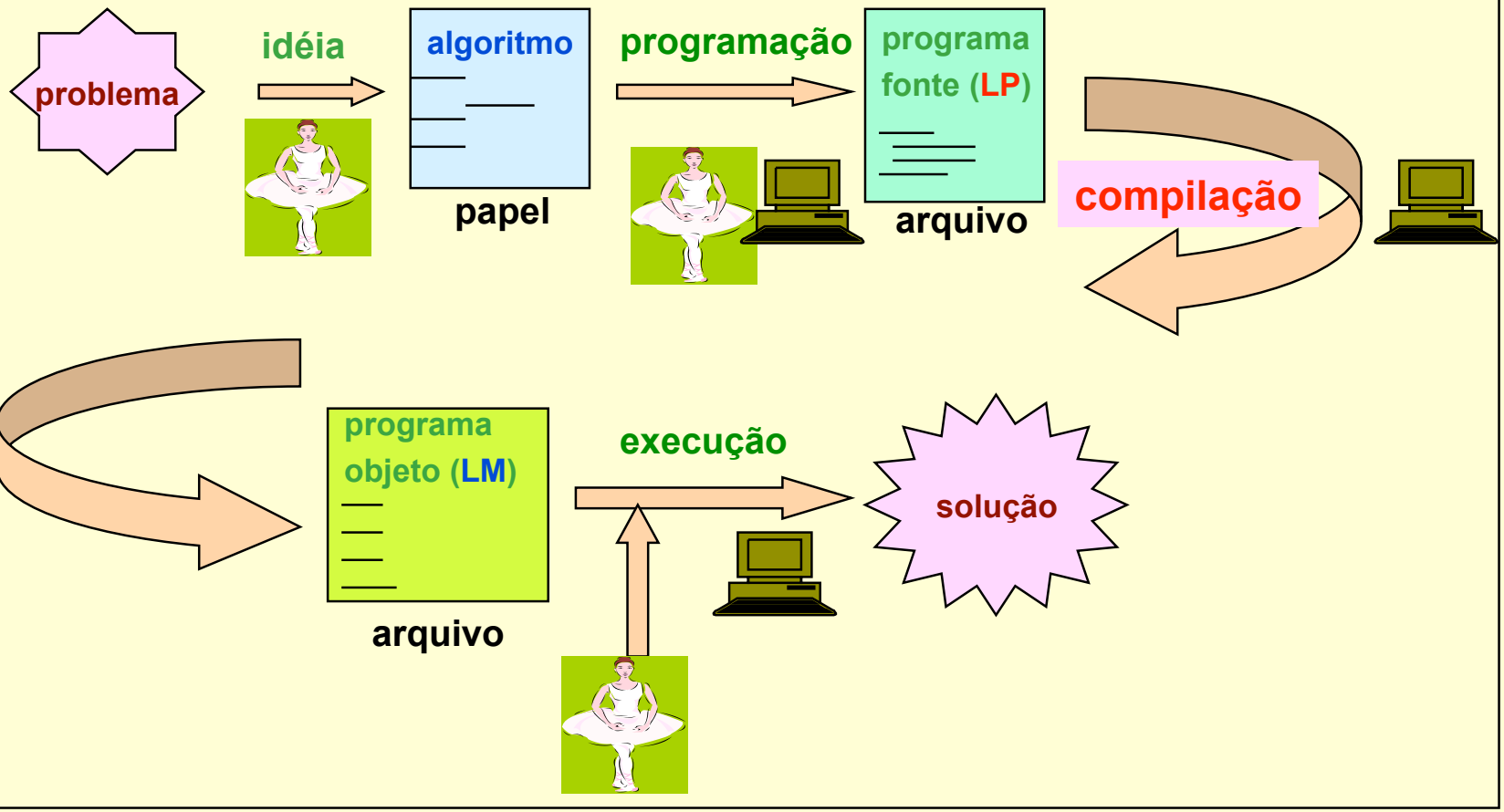
Existem milhares de LPs:

- FORTRAN: científica, mais antiga
- ALGOL, **C**, PASCAL: estruturadas, generalistas
- **C++**, C#, JAVA : lidam com tecnologia de objetos
- LISP, PROLOG: voltadas para IA
- . . . (muitas outras)

Algoritmos

... e compilação

O processo de compilação/edição/execução:



Algoritmos

... e compilação

- **Codificação/compilação/execução:**
  - permite **executar** um **algoritmo** em um **computador**
    - obtém **solução** para problemas
    - rapidamente (?!)

- **Algoritmo:**
  - **centro** do universo
  - demais componentes são **acessórios**
  - "Ciências da Computação": estudo de **algoritmos**
  - "Inteligência de um computador": algoritmos programados

Algoritmos ... resolvem qualquer problema?

Questão:

Dado um problema **P**, sempre haverá um **algoritmo** que resolva **P** corretamente?

- **P** deve ser um problema prático, fácil de enunciar
  - ordenar um vetor de números,
  - calcular produto de matrizes, . . .

- O **algoritmo A** que resolve **P** deve funcionar corretamente em todas as (infinitas) entradas de **P**
  - todos os vetores, carregados com inteiros quaisquer
  - quaisquer duas matrizes de quaisquer dimensões compatíveis entre si

Algoritmos ... resolvem qualquer problema?

A Ciência da Computação tem a resposta para a questão

SURPRESA ! ! !

NÃO. Há **problemas** para os quais **NÃO EXISTE** algoritmos capazes de resolve-los corretamente.

Com mais tecnologia (computadores mais rápidos, mais memória) e dado tempo suficiente para rodar o programa, poderemos resolver esses problemas, no futuro, certo?

NÃO

Computador nenhum vai resolver esses problemas, nem hoje, nem amanhã, nem nunca; nem aqui, nem em Marte, em lugar algum; rodando qualquer programa por quanto tempo quiser (anos,séculos, ...)

Algoritmos ... resolvem qualquer problema?

Um problema indecidível (insolúvel) [Harel, "Computers Ltd."]:

Dado um conjunto finito T de ladrilhos quadrados:

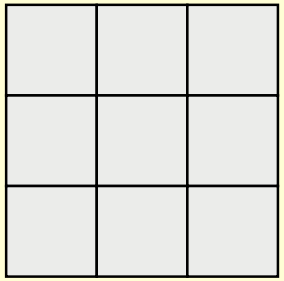
Problema: podemos ladrilhar qualquer grade quadrada com ladrilhos de tipo T, casando as cores das faces que se tocam? **SIM** ou **NÃO**?

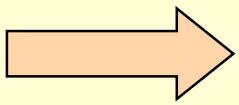
- pode usar quantos ladrilhos quiser, de cada tipo
- os ladrilhos não podem ser girados

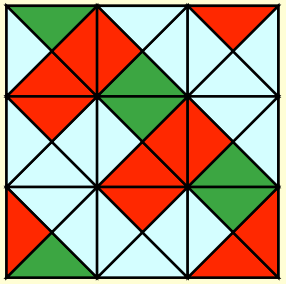
Algoritmos

... resolvem qualquer problema?

Exemplo 1:



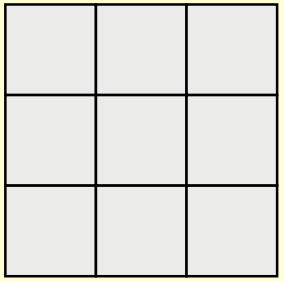


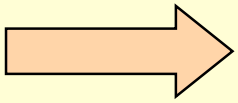


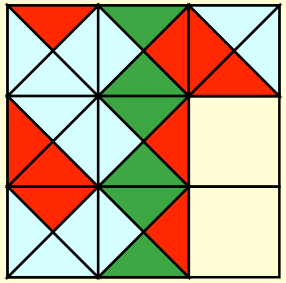
consegue para toda região do plano

SIM!

Exemplo 2:







todas as outras possibilidades falham nessa região

NÃO!

Algoritmos ... resolvem qualquer problema?

Problema de ladrilhar toda região do plano:

NENHUM computador **JAMAIS** vai conseguir resolver esse problema, nem agora, nem nunca, nem com **QUALQUER** melhoria de tecnologia, nem com **QUALQUER** tamanho de memória, nemcom **QUALQUER** tempo de execução

Podemos demonstrar isso, matematicamente!

Algoritmos ... resolvem qualquer problema?

Um problema decidível (solúvel) [Harel, "Computers Ltd."]:

Dado um conjunto finito T de ladrilhos quadrados:

Dadas duas posições ("I" e "F") na grade infinita do plano

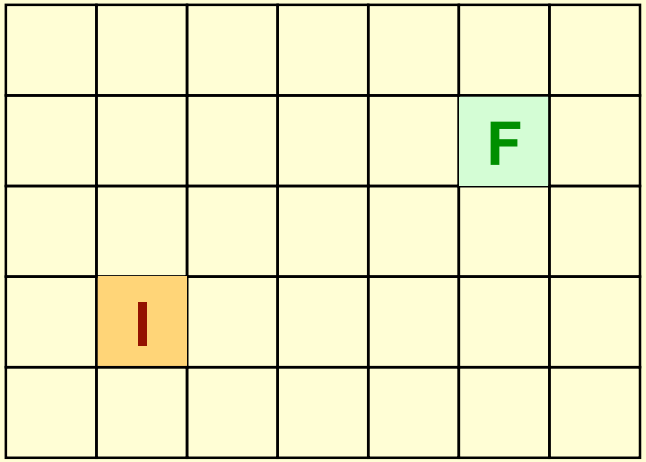
Problema: podemos ladrilhar um caminho na grade, partindo de "I" e chegando em "F", com ladrilhos de tipo T, e casando as cores das faces que se tocam? **SIM** ou **NÃO**?

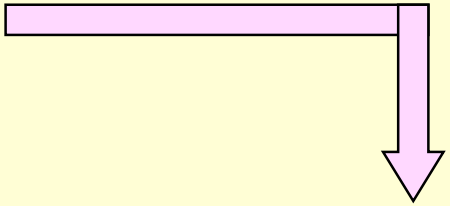
- pode usar quantos ladrilhos quiser, de cada tipo
- os ladrilhos não podem ser girados

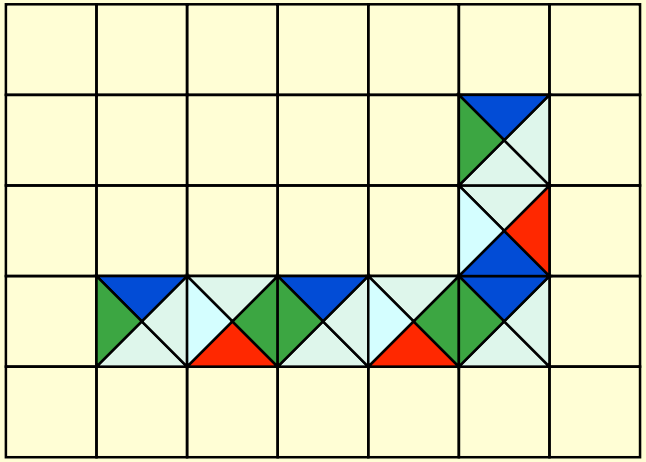
Algoritmos

... resolvem qualquer problema?

Exemplo:







Com **esses** ladrilhos

Com **essas** posições **I** e **F**

SIM!

Algoritmos ... resolvem qualquer problema?

Problema de ladrilhar um caminho entre duas posições:

EXISTE um **algoritmo** que **decide se há**, ou **se não há**, o caminho entre as duas posições dadas, usando ladrilhos do conjunto dado

Podemos **exibir** o **algoritmo** e **mostrar** sua **correção** e **terminação**, não importa qual o conjunto **T** dado e não importa quais as posições **I** e **F** dadas

Algoritmos ... resolvem qualquer problema?

Ladrilhar caminhos (levem. modificado) [Harel, "Computers Ltd."]:

Dado um conjunto finito T de ladrilhos quadrados:

Dadas duas posições ("I" e "F") na grade infinita do **SEMI-plano**

Problema: podemos ladrilhar um caminho na grade, partindo de "I" e chegando em "F", com ladrilhos de tipo T, e casando as cores das faces que se tocam? **SIM** ou **NÃO**?

- pode usar quantos ladrilhos quiser, de cada tipo
- os ladrilhos não podem ser girados

Não é permitido que o caminho cruze uma dada linha horizontal, que divide o plano em dois semi-planos

Algoritmos ... resolvem qualquer problema?

Problema de ladrilhar caminhos no semi-plano:

O problema, agora, torna-se INDECIDÍVEL!

NENHUM computador **JAMAIS** vai conseguir **resolver** esse problema, nem agora, . . .

Podemos demonstrar isso, matematicamente!